Software Design, Modelling and Analysis in UML

Lecture 12: Core State Machines II

2016-12-13

Prof. Dr. Andreas Podelski, Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany

Content

- Actions
  - transformer:
    - send message
    - create/destroy: later

- Labelled Transition System

- Transitions of UML State Machines
  - discard event,
  - dispatch event,
  - continue RTC,
  - environment interaction,
  - error condition.

- Example Revisited
**Definition.**
Let $\Sigma_D$ the set of system configurations over some $S_0, D_0, Eth$.

We call a relation
$$t \subseteq \left( \mathcal{H}(\mathcal{C}) \times (\Sigma_D \times Eth) \right) \times (\Sigma_D \times Eth)$$

a (system configuration) **transformer**.

**Example:**
- $t[u_x](\sigma, \epsilon) \subseteq \Sigma_D \times Eth$ is
  - the set (!) of the system configurations
  - which may result from object $u_x$
  - executing transformer $t$.  
- $t_{\text{skip}}[u_x](\sigma, \epsilon) = \{ (\sigma, \epsilon) \}$
- $t_{\text{create}}[u_x](\sigma, \epsilon)$ : add a previously non-alive object to $\sigma$
Observations

- In the following, we assume that
  - each application of a transformer \( t \)
  - to some system configuration \((\sigma, \varepsilon)\)
  - for object \( u_x \)
  is associated with a set of observations
    \[
    Obs_t[u_x](\sigma, \varepsilon) \in 2^{(\mathcal{P}(\mathcal{S}) \cup \{\ast, +\}) \times \mathcal{P}(\varepsilon)}.
    \]
- An observation
  \[
  (u_e, u_{dst}) \in Obs_t[u_x](\sigma, \varepsilon)
  \]
  represents the information that,
  as a "side effect" of object \( u_x \) executing \( t \) in system configuration \((\sigma, \varepsilon)\),
  the event \( u_e \) has been sent to \( u_{dst} \).

Special cases: creation ("\ast") / destruction ("\ast\ast").

A Simple Action Language

In the following we use

\[
\mathit{Act}_\mathcal{S} = \{\text{skip}\}
\]

\[
\cup \{\text{update}(expr_1, v, expr_2) \mid expr_1, expr_2 \in \mathit{Expr}_\mathcal{S}, v \in \mathit{atr}\}
\]

\[
\cup \{\text{send}(E(expr_1, ..., expr_n), expr_{dst}) \mid expr_1, expr_{dst} \in \mathit{Expr}_\mathcal{S}, E \in \mathcal{E}\}
\]

\[
\cup \{\text{create}(C, expr, v) \mid C \in \mathcal{C}, expr \in \mathit{Expr}_\mathcal{S}, v \in V\}
\]

\[
\cup \{\text{destroy}(expr) \mid expr \in \mathit{Expr}_\mathcal{S}\}
\]

and OCL expressions over \( \mathcal{S} \) (with partial interpretation) as \( \mathit{Expr}_\mathcal{S} \).
**Transformer: Skip**

<table>
<thead>
<tr>
<th>abstract syntax</th>
<th>concrete syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>skip</td>
<td>skip</td>
</tr>
</tbody>
</table>

**intuitive semantics**

do nothing

**well-typedness**

.//

**semantics**

\[ t_{\text{skip}}[u_2](\sigma, e) = \{(\sigma, e)\} \]

**observables**

\[ \text{Obs}_{\text{skip}}[u_2](\sigma, e) = \emptyset \]

**(error) conditions**

Not defined.

---

**Transformer: Update**

<table>
<thead>
<tr>
<th>abstract syntax</th>
<th>concrete syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>update(expr_1, v, expr_2)</td>
<td>expr_1, v \Rightarrow expr_2</td>
</tr>
</tbody>
</table>

**intuitive semantics**

Update attribute \( v \) in the object denoted by \( expr_1 \) to the value denoted by \( expr_2 \).

**well-typedness**

\( expr_1 : T_C \) and \( v : T \in \text{attr}(C) \); \( expr_2 : T \); \( expr_1, expr_2 \) obey visibility and navigability.

**semantics**

\[ t_{\text{update}}(expr_1, v, expr_2)[u_2](\sigma, e) = \{(\sigma', \sigma)\} \]

where \( \sigma' = \sigma[u \mapsto \sigma(v)](u' \mapsto I[expr_2](\sigma, u_2)) \) with \( u = I[expr_1](\sigma, u_2) \) (object denoted by \( expr_1 \) not defined).

**observables**

\[ \text{Obs}_{\text{update}}(expr_1, v, expr_2)[u_2] = \emptyset \]

**error conditions**

Not defined if \( I[expr_1](\sigma, u_2) \) or \( I[expr_2](\sigma, u_2) \) not defined.
**Update Transformer Example**

**SM C:**

\[ s_1 : x := x + 1 \]

\[ s_2 : \]

\[ \text{update}(\text{expr}_1, v, \text{expr}_2)[u_s](\sigma, \varepsilon) = (\sigma' = \sigma[u \mapsto \sigma(u)[v \mapsto I[\text{expr}_2](\sigma, u_s)]], \varepsilon), u = I[\text{expr}_1](\sigma, u_s) \]

---

**Transformer: Send**

**abstract syntax**

\[
\text{send}(E(\text{expr}_1, \ldots, \text{expr}_n), \text{expr}_{\text{dst}})
\]

**concrete syntax**

\[
E(\text{expr}_1, \ldots, \text{expr}_n)(\text{expr}_{\text{dst}})
\]

**intuitive semantics**

Object \( u_x : C \) sends event \( E \) to object \( \text{expr}_{\text{dst}} \), i.e. create a fresh signal instance, fill in its attributes, and place it in the ether.

**well-typedness**

\( E \in \delta \); \( \text{attr}(E) = \{ v_1 : T_1, \ldots, v_n : T_n \}; \text{expr}_i : T_i, 1 \leq i \leq n; \)

\( \text{expr}_{\text{dst}} : T_{\text{D}}, C, D \in \mathcal{C} \setminus \delta; \)

all expressions obey visibility and navigability in \( C \)

**semantics**

\( (\sigma', \varepsilon') \in t_{\text{send}}(E(\text{expr}_1, \ldots, \text{expr}_n), \text{expr}_{\text{dst}})[u_s](\sigma, \varepsilon) \)

if \( \sigma' = \sigma \uplus \{ u \mapsto \{ v_i \mapsto d_i \mid 1 \leq i \leq n \} \}; \varepsilon' = \varepsilon \uplus (\text{expr}_{\text{dst}}, u_E); \)

if \( u_{\text{dat}} = I[\text{expr}_{\text{dat}}](\sigma, u_s) \in \text{dom}(\sigma); \ d_i = I[\text{expr}_i](\sigma, u_s) \) for \( 1 \leq i \leq n; \)

\( u_E \in \mathcal{D}(E) \) a fresh identity, i.e. \( u_E \notin \text{dom}(\sigma) \).

and where \( (\sigma', \varepsilon') = (\sigma, \varepsilon) \) if \( u_{\text{dat}} \notin \text{dom}(\sigma); \text{ sending } u_x \text{ to } \text{expr}_{\text{dst}} \text{ from } u_x \text{ does nothing} \)

**observables**

\( \text{Obs}_{\text{send}}[u_s] = \{ (u_E, u_{\text{dat}}) \} \)

**error conditions**

\( I[\text{expr}_i](\sigma, u_s) \) not defined for any \( \text{expr} \in \{ \text{expr}_{\text{dat}}, \text{expr}_1, \ldots, \text{expr}_n \} \)
**Send Transformer Example**

\[ SM_C: \]

\[ \sigma: \]

\[ u_1: C \]

\[ x = 5 \]

\[ \tau: \]

\[ v_2: C \]

\[ u = 6 \]

\[ \varepsilon: \]

\[ \tau: \]

\[ v_1: C \]

\[ \tau: \]

**Sequential Composition of Transformers**

- **Sequential composition** \( t_1 \circ t_2 \) of transformers \( t_1 \) and \( t_2 \) is canonically defined as

\[ (t_2 \circ t_1)[u_x](\sigma, \varepsilon) = t_2[u_x](t_1[u_x](\sigma, \varepsilon)) \]

with observation

\[ \text{Obs}_{(t_2 \circ t_1)}[u_x](\sigma, \varepsilon) = \text{Obs}_{t_1}[u_x](\sigma, \varepsilon) \cup \text{Obs}_{t_2}[u_x](t_1(\sigma, \varepsilon)). \]

- **Clear**: not defined if one the two intermediate “micro steps” is not defined.
**Observation:** our transformers are in principle the denotational semantics of the actions/action sequences. The trivial case, to be precise.

**Note:** with the previous examples, we can capture
- empty statements, skips,
- assignments,
- conditionals (by normalisation and auxiliary variables),
- create/destroy (later),
but not possibly diverging loops.

**Our (Simple) Approach:** if the action language is, e.g. Java, then (syntactically) forbid loops and calls of recursive functions.

**Other Approach:** use full blown denotational semantics.

No show-stopper, because loops in the action annotation can be converted into transition cycles in the state machine.

---

**Course Map**

```plaintext
\varphi \in OCL
F = (\mathcal{F}, \mathcal{C}V, atr), SM
M = (\Sigma_{\mathcal{F}, A_{\mathcal{F}}, SM})
B = (Q_{SD}, q_0, \delta_{SD}, F_{SD})
\pi = (\sigma_0, \varepsilon_0) \rightarrow (\sigma_1, \varepsilon_1) \rightarrow \cdots 
\omega = ((\sigma_i, cons_i, Snd_i))_{i \in \mathbb{N}}
G = (N, E, f)
\sigma_0 \xrightarrow{cons_0, Snd_0} \sigma_1 \xrightarrow{cons_1, Snd_1} \cdots
```

---

\( \mathcal{C} \subseteq \mathbb{N} \times \mathbb{N} \subseteq \mathcal{E} \times \mathbb{N} \times \mathbb{N} \)
Definition. Let $A$ be a set of labels and $S$ a (not necessarily finite) set of states. We call

$$\not\rightarrow \subseteq S \times A \times S$$

a (labelled) transition relation.

Let $S_0 \subseteq S$ be a set of initial states. A (finite or infinite) sequence $s_0 \not\rightarrow a_0 s_1 \not\rightarrow a_1 s_2 \not\rightarrow a_2 \ldots$ with $s_i \in S, a_i \in A$ is called computation of the labelled transition system $(S, A, \not\rightarrow, S_0)$ if and only if

- initiation: $s_0 \in S_0$
- conclusion: $(s_i, a_i, s_{i+1}) \in \not\rightarrow$ for $i \in \mathbb{N}_0$. 

Transition Relation, Computation
Active vs. Passive Classes/Objects

- **Note**: From now on, for simplicity, assume that all classes are active.

We'll later briefly discuss the Rhapsody framework which proposes a way how to integrate non-active objects.

- **Note**: The following RTC “algorithm” follows Harel and Gery (1997) (i.e. the one realised by the Rhapsody code generation) if the standard is ambiguous or leaves choices.

From Core State Machines to LTS

Definition. Let \(\mathcal{S}_0 = (\mathcal{F}_0, \mathcal{C}_0, V_0, atr_0, E)\) be a signature with signals (all classes in \(\mathcal{C}_0\) active), \(\mathcal{D}_0\) a structure of \(\mathcal{S}_0\), and \((Eth, \text{ready}, \oplus, \ominus, [\cdot])\) an ether over \(\mathcal{S}_0\) and \(\mathcal{D}_0\). Assume there is one core state machine \(M_C\) per class \(C \in \mathcal{C}\).

We say, the state machines induce the following labelled transition relation on states \(S := (\Sigma \times Eth) \cup \{\#\}\) with labels \(A := 2^{\mathcal{G}(E)} \times 2^{(\mathcal{G}(E) \cup \{\#\}) \times \mathcal{G}(C)} \times \mathcal{G}(C)\):

- \((\sigma, \varepsilon) \xrightarrow{(cons, \text{Stop})} (\sigma', \varepsilon')\) if and only if
  1. an event with destination \(u\) is discarded, or
  2. an event is dispatched to \(u\), i.e. stable object processes an event, or
  3. run-to-completion processing by \(u\) continues, i.e. object \(u\) is not stable and continues to process an event, or
  4. the environment interacts with object \(u\), or
  5. an error condition occurs during consumption of \(\text{cons}\), or

- \(s \xrightarrow{(\text{cons}, \emptyset)} \#\) if and only if
  1. an error condition occurs during consumption of \(\text{cons}\), or
  2. \(s = \#\) and \(\text{cons} = \emptyset\).
(i) Discarding An Event

\[(\sigma, \varepsilon) \xrightarrow{\text{(cons,Snd)}}_{u} (\sigma', \varepsilon')\]

if

\[
\begin{align*}
\text{conditions on } (\sigma, \varepsilon) \\
\text{and} \\
\text{conditions on } (\sigma', \varepsilon')
\end{align*}
\]

• an \(E\)-event (instance of signal \(E\)) is ready in \(\varepsilon\) for object \(u\) of a class \(C\), i.e. if

\[
u \in \text{dom}(\sigma) \cap \mathcal{P}(C) \land \exists \ u_E \in \mathcal{P}(E) : u_E \in \text{ready}(\varepsilon, u)
\]

• \(u\) is stable and in state machine state \(s\), i.e. \(\sigma(u)(\text{stable}) = 1\) and \(\sigma(u)(\text{st}) = s\),

• but there is no corresponding transition enabled (all transitions incident with current state of \(u\) either have other triggers or the guard is not satisfied)

\[
\forall (s, F, \text{expr}, \text{act}, s') \in \rightarrow (SM_C) : F \neq E \lor I[\text{expr}](\sigma, u) = 0
\]

and

• in the system configuration, stability may change, \(u_E\) goes away, i.e.

\[
\sigma' = \sigma[u.\text{stable} \mapsto b] \setminus \{u_E \mapsto \sigma(u_E)\}
\]

where \(b = 0\) if and only if there is a transition with trigger \(\_\_\_\) enabled for \(u\) in \((\sigma', \varepsilon')\).

• the event \(u_E\) is removed from the ether, i.e.

\[
\varepsilon' = \varepsilon \ominus u_E,
\]

• consumption of \(u_E\) is observed, i.e.

\[
\text{cons} = \{u_E\}, \quad \text{Snd} = \emptyset.
\]
Example: Discard

\[ \begin{align*}
&G[x > 0]/x := y \\
&H/z := y/x \\
&[x > 0]/x := x - 1; n! J
\end{align*} \]

\[ \begin{align*}
&\sigma : C \\
&x = 1, z = 0, y = 2 \\
&st = s_1 \\
&stable = 1
\end{align*} \]

\[ \begin{align*}
&\{u_3\}, \emptyset \\
&x = z = 0, y = 1 \\
&st = s_2 \\
&stable = 0
\end{align*} \]

(ii) Dispatch

\[ (\sigma, \varepsilon) \xrightarrow{\text{(cons, Snd)}} (\sigma', \varepsilon') \]

if

- \( u \in \text{dom}(\sigma) \cap \mathcal{D}(C) \) and \( u \in \mathcal{D}(E) \) and \( u \in \text{ready}(\varepsilon, u) \)
- \( \forall (s, F, expr, act, s') \in (SM_C) : F \neq E \land I[expr][\sigma, u] = 0 \)
- \( \sigma(u)(\text{stable}) = 1, \sigma(u)(st) = s \)
- \( \sigma'(u) \) is \( \text{enabled} \), i.e.

\[ \exists (s, F, expr, act, s') \in (SM_C) : F = E \land I[expr](\sigma, u) = 1 \]

where \( \hat{\sigma} = \sigma[u.\text{params} \mapsto u_E] \).

and

- \( (\sigma', \varepsilon') \) results from applying \( t_{act} \) to \( (\sigma, \varepsilon) \) and removing \( u_E \) from the ether, i.e.

\[ \sigma'' \in t_{act}(\sigma, \varepsilon, \sigma'' \in \mathcal{D}(C) \}

where \( b \) depends (see (i))

- Consumption of \( u_E \) and the side effects of the action are observed, i.e.

\[ \text{cons} = \{u_E\}, \quad \text{Snd} = \text{Obs}_{\text{act}}[u](\sigma, \varepsilon \oplus u_E) \]
(iii) Continue Run-to-Completion

\[
(\sigma, \varepsilon) \xrightarrow{\text{cons, Snd}} (\sigma', \varepsilon')
\]

if

- there is an unstable object \( u \) of a class \( C \), i.e.
  \[
  u \in \text{dom}(\sigma) \cap D(C) \land \sigma(u)(\text{stable}) = 0
  \]

- there is a transition without trigger enabled from the current state \( s = \sigma(u)(st) \), i.e.
  \[
  \exists (s, \_ , \text{expr}, \text{act}, s') \in \rightarrow(SMC) : \Pi[\text{expr}](\sigma, u) = 1
  \]

and

- \((\sigma', \varepsilon')\) results from applying \( t_{\text{act}} \) to \((\sigma, \varepsilon)\), i.e.
  \[
  (\sigma'', \varepsilon') \in t_{\text{act}}[\sigma, \varepsilon], \quad \sigma' = \sigma''[u \mapsto s', u, \text{stable} \mapsto b]
  \]

where \( b \) depends as before.

- Only the side effects of the action are observed, i.e.
  \[
  \text{cons} = \emptyset, \quad \text{Snd} = \text{Obs}_{t_{\text{act}}}[u](\sigma, \varepsilon).
  \]
(iv) Environment Interaction

Assume that a set $\mathcal{E}_{env} \subseteq \mathcal{E}$ is designated as environment events and a set of attributes $V_{env} \subseteq V$ is designated as input attributes.

Then

$$\begin{align*}
(\sigma, \varepsilon) \xrightarrow{(cons, \text{Sort})}_{env} (\sigma', \varepsilon')
\end{align*}$$

if either (!)

- an environment event $E \in \mathcal{E}_{env}$ is spontaneously sent to an alive object $u \in \text{dom}(\sigma)$, i.e.

$$\sigma' = \sigma \cup \{ u_E \mapsto \{ v_i \mapsto d_i \mid 1 \leq i \leq n \} \}$$

where $u_E \not\in \text{dom}(\sigma)$ and $\text{atr}(E) = \{ v_1, \ldots, v_n \}$.

- Sending of the event is observed, i.e. $\text{cons} = \emptyset$, $\text{Sort} = \{ u_E, \}$.

or

- Values of input attributes change freely in alive objects, i.e.

$$\forall v \in V \forall u \in \text{dom}(\sigma) : \sigma'(u)(v) \neq \sigma(u)(v) \implies v \in V_{env}.$$ 

and no objects appear or disappear, i.e. $\text{dom}(\sigma') = \text{dom}(\sigma)$.

- $\varepsilon' = \varepsilon$. 

Example: Continue

$SM_C$: $SM_C: G[x > 0]/x := y$

$H/z := y/x$

$\sigma$: $\sigma: x = 2, z = 0, y = 2$

$n$ $\sigma'$

$\varepsilon$: $\varepsilon: \sigma'_1, \{ (\varepsilon, \varepsilon) \}$

$\varepsilon'$

- $u \in \text{dom}(\sigma) \cap \mathcal{D}(C)$

$u_E \in \mathcal{D}(E), u_E \in \text{ready}(\varepsilon, u)$

$\forall (s, F, expr, act, s') \in \rightarrow (SM_C)$:

$F \neq E \lor [\text{expr}]([\sigma, u]) = 0$

$\sigma'(\text{stable}) = 1, \sigma(u)(\text{st}) = n,$

$\sigma' = \sigma[u, \text{stable} \mapsto b] \setminus \{ u_E \mapsto \sigma(u_E) \}$

$\varepsilon' = \varepsilon \oplus u_E$

$\text{cons} = \{ u_E \}, \text{Sort} = \emptyset$

- $\varepsilon''$
(v) Error Conditions

\[ s \xrightarrow{\text{(cons, SND)}} u \xrightarrow{\#} \]

if, in (i), (ii), or (iii),

- \( I[\text{expr}] \) is not defined for \( \sigma \) and \( u \), or
- \( t_{act}[u] \) is not defined for \((\sigma, \varepsilon)\),

and

- \( \text{cons} = \emptyset \), and \( \text{Snd} = \emptyset \).

Examples:

- \( E[x/0]/\text{act} \) \xrightarrow{\#} \[ s \]
- \( E[\text{true}]/\text{act} \) \xrightarrow{\#} \[ s \]
- \( E[\text{expr}]/x := x/0 \) \xrightarrow{\#} \[ s \]
**Example: Error Condition**

\[ |x > 0|/x := x - 1; n!J \]

\[ G[x > 0]/x := y \]

\[ H/z := y/x \]

<table>
<thead>
<tr>
<th>( n )</th>
<th>( \sigma )</th>
<th>( \varepsilon )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x = 0, z = 0, y = 27 )</td>
<td>( st = s_2, stable = 1 )</td>
<td></td>
</tr>
<tr>
<td>( \varepsilon : (c, u_H : H) )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- \( u \in \text{dom}(\sigma) \cap \mathcal{F}(C) \)
- \( u_E \in \mathcal{F}(E), u_E \in \text{ready}(\varepsilon, u) \)
- \( \forall (s, F, \text{expr}, act, s') \in \rightarrow(SMC) : F \neq E \lor I[\text{expr}](\sigma, u) = 0 \)
- \( \sigma(u)(\text{stable}) = 1, \sigma(u)(st) = s, \)
- \( \sigma'(u, \text{stable} \mapsto b) \setminus \{u_E \mapsto \sigma(u_E)\} \)
- \( \varepsilon' = \varepsilon \cap u_E \)
- \( \text{cons} = \{u_E\}, \ Snd = \emptyset \)

**Example Revisited**

\[ E[x \neq 0]/x := x + 1; n!F \]

\[ F/z := 0 \]

\[ /n := 0 \]

\[ \varepsilon : (c, u_H : H) \]

<table>
<thead>
<tr>
<th>Nr</th>
<th>( x )</th>
<th>( n )</th>
<th>( s )</th>
<th>( st )</th>
<th>( p )</th>
<th>( st )</th>
<th>( \varepsilon )</th>
<th>( \text{rule} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>27</td>
<td>5_D</td>
<td>( s_1 )</td>
<td>1</td>
<td>( s_1 )</td>
<td>1</td>
<td>( (3_F, 1_C), (2_E, 1_C) )</td>
<td></td>
</tr>
</tbody>
</table>
• State Machines induce a labelled transition system.

• There are five kinds of transitions in the LTS:
  • discard: no matching state machine edge enabled, may change stability.
  • dispatch: a matching state machine edge is taken, i.e. actions are executed (according to transformers),
  • continue: a state machine edge without signal-trigger is enabled, and is taken,
  • environment interaction: dedicated environment signals are injected into the event pool,
  • error condition: a designated error state is assumed, maybe due to undefined action transformers.

• For now, we assume that all classes are active, thus steps of objects may interleave.

References
References

